View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Servicio de Difusión de la Creación Intelectual

Comparative Study of Trajectory Metaheuristics for the Resolution of Scheduling Problem of Unrestricted Parallel Identical Machines

Claudia R. Gatica, Susana C. Esquivel, and Guillermo M. Leguizamón

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis
Ejército de Los Andes 950 - Local 106
(5700) - San Luis - Argentina
Tel: (0266) 4420823 / Fax: (0266) 4430224
{crgatica,esquivel,legui}@unsl.edu.ar

Abstract. In this paper we present a comparative study of four trajectory metaheuristics or single-solution based metaheuristics (S-meta heuristics): Iterated Local Search (ILS), Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Search (VNS) and Simulated Annealing (SA). The metaheuristics were used to minimize the *Maximum Tardiness* (T_{max}) for unrestricted parallel identical machine scheduling (P_m) problem, which is considered as NP-Hard problem. The results obtained through experimentation show that SA was the best behaved.

Keywords: Iterated Local Search, Greedy Randomized Adaptive Search Procedure, Variable Neighborhood Search, Simulated Annealing, Parallel Machines Scheduling, Maximum Tardiness, Statistical Nonparametric Tests.

1 Introduction

The unrestricted parallel identical machines problem P_m considered in this paper consists of scheduling n jobs on m identical parallel machines P_m to minimize the T_{max}. There are no constraints in the assignement of jobs to machines, therefore the problem is described by a $(P_m||T_{max})$. This problem belongs to more basic model of P_m which is NP-hard, even when $m = 2$ [13]. The P_m are representative of many real world problems. In such systems is usual to minimize objective functions based on the due dates, such as the T_{max}. Metaheuristics have been used to solve similar problems that we are concerned. For instance in [17], [13] a set of dispatching rules and heuristics are presented. In a related work [15] the P_m was solved with SA and GRASP algorithms to minimize the maximum completion time (i.e., the makespan). In [6] SA was used to solve P_m with sequence-dependent setup times to minimize the makespan. The VNS algorithm was developed in [12] to minimize the weighted number of late jobs with release dates and in [9] to minimize the makespan. The ILS algorithm was

presented in [1] to solve P_m with sequence-dependent setup times and unequal ready times to minimize total weighted number of tardy jobs.

The remainder of the paper is organized as follows. Section 2 introduces the P_m problem. In Section 3, the trajectory metaheuristics are presented. The experimental design is described in Section 4. In Section 5, the results are analyzed. Section 6 provides our conclusions and future work.

2 Unrestricted Parallel Machine Scheduling Problem

The formal notation used in the literature [13] for the scheduling problem that we are dealing is a triplet: (P_m || T_{max}). The first field describes the machine environment P_m, the second one contains the restrictions, we note that our problem is unrestricted, therefore this field is empty, and the third provides the objective function T_{max} to be optimized. This scheduling problem can be stated as follows: there are n jobs to be processed without interruption on some of the m identical machines belonging to the system P_m; each machine can process not more than one job at a time. Job *j (j=1,2,...n)* is made available for processing at time zero. It requires an uninterrupted positive processing time p_j on a machine and it has a due date d_j by which it should ideally be finished. For a given processing order of the jobs (schedule), the earliest completion time C_j and the maximum delay time $T_j = \{C_j$ - $d_j, 0\}$ of the job j can be easily estimated. The problem consists in finding an optimum schedule objective value. The objective to be minimized is:

$$MaximumTardiness : T_{max} = max_j(T_j)$$

The problems related to the due dates have received considerable attention from a practical and theoretical point of view. Besides, they are considered as NP-Hard when $2 \leq m \leq n$ [13].

3 Description of Trajectory Metaheuristics

Single-solution based metaheuristics (S-metaheuristics) improve a single solution. They could be viewed as *walks* through neighborhoods or search trajectories through the search space of the problem at hand [5]. The walks (or trajectories) are performed by iterative procedures that move from the current solution to another one in the search space. S-metaheuristics show their efficiency in tackling various optimization problems in different domains.

3.1 BLS

Basic Local Search is the oldest and simplest metaheuristic. It starts at a given initial solution. At each iteration, the heuristic replaces the current solution by a neighbor that improves the objective function. The search stops when all candidate neighbors are worse than the current solution, meaning a local optimum

is reached. Algorithm 1 describes this process [5]. BLS metaheuristic is included here because it is invoked by the other metaheuristics, but it is not involved in our comparative analysis.

Algorithm 1 Basic Local Search (BLS)

```
s=s_0 {initial solution}
while {not Termination Criterion} do
   Generate(N(s)) {Generation of candidate neighbors}
   if {There is no better neighbor in N(s)} then
      Stop
   else
      s'=s {Select a better neighbor in N(s)}
   end if
end while
{Output: Final solution found (local optima)}
```

3.2 ILS

Iterated local search may be used to improve the quality of successive local optima. In multistart local search, the initial solution is always chosen randomly and then is unrelated to the generated local optima. ILS improves the classical multistart local search by perturbing the local optima and reconsidering them as initial solutions. Algorithm 2 shows the pseudocode of ILS [5].

Algorithm 2 Iterated Local Search (ILS)

```
s*=Local-Search(s_0) {Apply a local search algorithm}
repeat
   s'= Perturb(s*) {Perturb the obtained local optima}
   s''=Local-Search(s') {Apply a local search algorithm on the perturbed solution}
   s*= Accept(s*, s'') {accepting criteria}
until Stopping criteria
{Output: Best solution found}
```

3.3 GRASP

GRASP metaheuristic is an iterative greedy heuristic to solve combinatorial optimization problems. Each iteration of the GRASP algorithm contains two steps: construction and local search. In the construction step, a feasible solution is built using a randomized greedy algorithm, then a local search heuristic is applied to the solution constructed in the previous step. GRASP pseudocode is displayed in Algorithm 3 [5].

Algorithm 3 Greedy Random Adaptive Search Procedure (GRASP)

{**Input**: Number of iterations}
repeat
  s=$Greedy(seed)$ {Apply a randomized greedy heuristic}
  s'=Local-search(s) {Apply a local search algorithm}
until Stopping criteria {a given number of iterations}
{**Output**: Best solution found}

3.4 VNS

The basic idea of VNS is to successively explore a set of predefined neighborhoods to provide a better solution. It explores either at random or systematically a set of neighborhoods to get different local optima. VNS exploits the fact that using various neighborhoods in local search may generate different local optima and that the global optima is a local optima for a given neighborhood. Indeed, different neighborhoods generate different landscapes. VNS pseudocode is given in Algorithm 4 [5].

Algorithm 4 Basic Variable Neighborhood Search (VNS)

{**Input**: a set of neighborhood structures N_k for $k = 1, ..., k_{max}$}
s=s_0 {Generate the initial solution}
repeat
  $k = 1$
  repeat
    Shaking() {Pick a random solution s' from the N_k(s)}
    s''=Local-search(s') {Apply a local search algorithm}
    if $s'' < s'$ **then**
      $s = s''$
      Local-search(s) {Continue to search with N_1(s), k=1}
    else
      $k = k + 1$
    end if
  until $k = k_{max}$
until Stopping Criteria
{**Output**: Best solution found.}

3.5 SA

SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure. The strength of the structure depends on the rate of cooling metals. If the initial temperature is not sufficiently high or a fast cooling is applied, imperfections (metastable states) are obtained. In this case, the cooling solid will not attain thermal equilibrium at each temperature. Strong crystals are grown from careful and slow cooling. The SA algorithm simulates the energy changes in a system subjected to a cooling process until it converges to an equilibrium state (steady frozen state). This scheme was developed in 1953 by Metropolis [14] and it is described in Algorithm 5 [5].

Algorithm 5 Simulated Annealing (SA)

$k = 0$ {Using for the Equilibrium condition}
$s = s_0$ {Generation of the initial solution}
$T = T_{max}$ {Starting temperature}
repeat
  repeat
    $k = k + 1$
    Generate a random neighbor s'
    $\Delta E = f(s') - f(s)$
    if $\Delta E \leq 0$ **then**
      $s = s'$
    else
      Acept s' with a probability $e^{\frac{-\Delta E}{T}}$
    end if
  until $mod(k, Markov\text{-}chain\text{-}length) == 0$ {Equilibrium condition}
  Update (T) {Temperature Update}
until Stopping Criteria
{**Output**: Best solution found.}

4 Description of Experiments

As it is not usual to find published benchmarks (known optimal values) for the unrestricted parallel identical machines scheduling problems, we work with some that were used by previous works such as [2], [3] and [4]. In those works, the problem instances were built based on selected data corresponding to weighted tardiness problems and they were taken from the OR-Library [7]. These data were the input for dispatching rules and conventional heuristics, and PARSIFAL [17], a software package provided by Morton and Pentico, were used to evaluate the instances problem by means of different heuristics, for example *EDD* (Earliest Due Date first) and *SPT* (Shortest Processing Time first). In this way, the benchmarks for problem instances for the T_{max} objective function were obtained. There are 20 problem instances of $n = 100$ jobs each, and they have an increasing identification number, meaning that as its identification number increases, so does its complexity. This is because they were generated with more high average tardiness factor TF. We use the following three performance metrics:

1. *Best*: it is the global best solution found in each run.
2. *Mean best (MBest)*: it is the mean value of *Best* throughout all runs.
3. *Ebest* $= ((best\ value - opt\text{-}val)/opt\text{-}val) * 100$: it is the percentage error of the best found solution when compared with the known or estimated (upper bound) optimum value *opt-val*. It gives a measure on how far the best solution is from that *opt-val*. When this value is negative, it means that the *opt-val* has been improved.

Before the optimization runs are started, we drive the designs for computer experiments to choose the best parameter values for each metaheuristic. There are two different design techniques described in [16]. The samples can be placed either on the boundaries, or in the interior of the design space. The former technique is used in the classical design of experiments (DOE) and the second is used by a more modern method called *Design and Analysis of Computer Experiments* (DACE). DACE assumes that the interesting features of the true model can be

found in the whole sample space. Therefore *space-filling* or exploratory design, which place a set of samples in the interior of the design space, are commonly used. Mckay et al (1979) proposed *Latin Hypercube Sampling* (LHS) as an alternative to the first proposed *Monte Carlo sampling*. LHS can be used to generate the design points for algorithm designs.

4.1 Parameter Settings

We used the statistical software Project R to generate design points for each metaheuristics. And thus, we obtained a LHD for each algorithm. Each desing point represents one configuration of parameters. Thus, for ILS, GRASP, VNS and SA we obtained a LHD of 20 points, one for each metaheuristic, and we did 20 experiments separately, because each metaheuristic has their proper design space. The configuration of parameters for BLS was included in ILS, VNS, and GRASP. The ranges of the parameters are given to generate the LHD samples, and they represent the dimensions of the design points. The Table 1 depicts the parameter ranges, where NS = Neighboorhood Size, OP = Operator Perturbation, NI = Number of Iterations, k=number of neighborhood structures, VNSI=Varible Neighboorhood Size, MCL=Markov Chain Length, CR=Cooling Rate, and IT = Initial Temperature.

Table 1. Parameter Ranges

Heurístic	NS	OP	NI	k	VNSI	MCL	CR	IT
ILS	[1,10]	[1,5]	[10e3,15e3]	–	–	–	–	–
GRASP	[1,10]	[1,5]	[10e3,15e3]	–	–	–	–	–
VNS	[1,10]	[1,5]	[10e3,15e3]	[1,20]	[1,30]	–	–	–
SA	–	[1,5]	[10e3,15e3]	–	–	[10,100]	[0,1]	[10e3,15e3]

We applied the Friedman test [10], [11], and [8], (Friedman two-way analysis of variances by ranks) which is a nonparametric similar of the parametric two-way analysis of variance. It can be used for answering the following question: is a set of k samples (where $k \geq 2$), do at least two of the samples which represent populations with different mean values?. The Friedman test is a multiple comparison test that aims to detect significant differences between the behavior of two or more algorithms.

Table 2. Parameter Settings

Heurístic	Config.	NS	OP	NI	k	VNSI	MCL	CR	IT
ILS	c_6	8	1	$10389 * 30$	–	–	–	–	–
GRASP	c_{10}	10	2	$11785 * 30$	–	–	–	–	–
VNS	c_9	9	1	$12400 * 30$	4	25	–	–	–
SA	c_{13}	–	5	$14717 * 30$	–	–	89	0.64	4039

For each metaheuristics, we applied the Friedman test and its results indicate us which parameter configurations are better than others. Theses configurations are shown in Table 2.

5 Analysis of results

All the metaheuristics have been run 30 times for each problem instance. Each run stop when the maximal number of objective function evaluations (300000) is achieved. In Table 3 we show the *Best* found by each metaheuristic and we can

Table 3. The Best achieved by each metaheuristic

Ins.	Bench.	ILS	GRASP	VNS	SA
1	**548**	839	907	888	557
6	**1594**	1933	1622	1850	**1574**
11	**2551**	2925	2744	2887	2552
19	**3703**	3965	3960	4095	3745
21	**5187**	5480	5387	5460	**5177**
26	**84**	1020	721	993	**84**
31	**1134**	2055	1836	2057	1145
36	**2069**	3026	2604	3065	2081
41	**3651**	4804	4080	4734	**3611**
46	**4439**	5029	4753	4827	4443
56	**617**	2456	1564	2290	621
61	**1582**	3389	3132	3141	1595
66	**2360**	3790	3450	3755	2372
71	**3786**	4955	4880	4848	3824
86	**1194**	3560	2860	3570	**1194**
91	**2204**	4083	3277	4170	2274
96	**3185**	4480	4255	4372	3259
111	**1365**	3896	3902	4305	1466
116	**2222**	4218	3405	4408	2307
121	**2999**	4693	4387	4670	3281
avg	**2323,7**	**3529,80**	**3186,30**	**3519,25**	**2358,10**

see that SA improves 3 values and achieves 2 of them. ILS, GRASP and VNS show, in a first analysis, similar mean.

The results indicate that SA had a better performance in all the problem instances, this is graphically ilustred in figure 1. There, the boxplots draw the values of *Ebest* metric, which represent the percentage error of the best found solution when compared with the known or estimated optimum value. The boxplot of SA is near to zero.

For the analysis of the results we use the Friedman test. In nonparametric statistics is common to use this test to determine the difference between more than two related samples.

In this study the related samples are the performance of the metaheuristics measured across the same data sets. The null hypothesis being tested is that all methods obtain similar results with nonsignificant differences.

The first step in calculating the test statistic is to convert the original results to ranks. Thus, the best performing algorithm should have the rank of 1, the

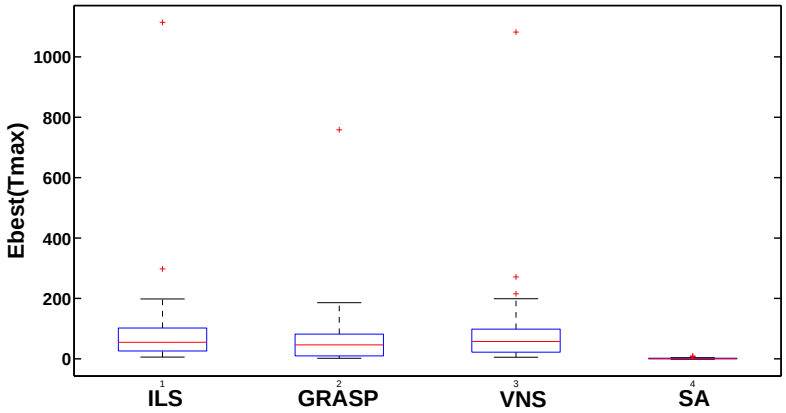

Fig. 1. Boxplots of Trajectory Metaheuristics.

second best rank 2, etc. In order to verify the null hypothesis, if the algorithms have similar behavior ranges should be equal.

In our analysis the Friedman test takes as input for each algorithm/problem pair the values of the metric *Mbest* (it is the mean value of *Best* value found by each metaheuristic throughout all runs).

Table 4. Rankings of the algorithms

Algorithm	Friedman (R_j)	Friedman Aligned	Quade
ILS	3.9000000000000012	57.64999999999999	3.9000000000000004
GRASP	2.150000000000001	39.2	2.0523809523809526
VNS	2.9499999999999993	54.64999999999999	3.047619047619047
SA	1.0000000000000002	10.499999999999998	1.0
statistic	54.42000000000013	14.575777663237545	136.01165501165502
P-value	$6.7591932051414E$-11	0.002217522627622426	$5.870461388944716E$-26

The test results are shown in Table 4. From the observation of the same we can reject the null hypothesis since the ranges differ, which indicates that there are significant differences between the algorithms. Also the heuristic that shows the best behavior is SA because it is the first in the ranking. Additionally we also apply the Friedman Aligned Ranks and Quade tests which offer a different way of ranking computation, but all of them located SA in the first place.

The main drawback of the previous tests is that they only detect significant differences over the whole multiple comparisons, being unable to establish proper comparison between some of the metaheuristics considered [8].

To compare the algorithms with each other and check if differences exist between them, we used a post-hoc procedure. The process is as follows: choose the heuristic that has shown better performance (in our case SA) and it is used as a control algorithm. Then each of the other heuristics used in the experimental study is compared to control heuristic, and a family of hypotheses is built, all referred to the control method. The application of a post-hoc test allows us to obtain p-values that determines the degree of rejection of each hypothesis, depending on a certain level of significance. The Table 5 displays the adjusted p-values obtained by the application of Holm Post-hoc test.

Table 5. The adjusted p-values of Holm Post-hoc test

Algorithm	Friedman	Friedman Aligned	Quade
ILS	$3.648555435089692E$-12	$4.188721955300531E$-10	$1.1096750436689544E$-6
VNS	$3.5673624347880124E$-6	$3.755266010513546E$-9	$6.624461201617061E$-4
GRASP	0.004848762721678939	$9.400147582069817E$-5	0.06505610894545293

In the comparison by pairs we can note a significant difference between VNS, ILS and SA since all p-values are less than 0.05. The above is not satisfied in the case of GRASP and Quade test, but considering the average values shown in Table 3 and also, that the p-value obtained for Quade is slightly higher than 0.05, we can say that SA overcomes the performance of GRASP.

6 Conclusions

We compare four trajectory metaheuristics: ILS, GRASP, VNS and SA to minimize the maximum tardiness for unrestricted parallel identical machine scheduling problem. Previously we realized a statistical study to determinate de best parameter values for each metaheuristic using the DACE method. Finally, we analyzed the results using different nonparametric statistical tests that allowed us to perform multiple comparisons between the algorithms to determine if any of the heuristics was better to solve the problem at hand. The tests shown that SA had a better performance. SA was then compared with each other metaheuristics, using the Holm Post-hoc test, in order to determine whether there were significant differences between them. The results showed that SA had indeed statistically significant differences with ILS and VNS but not so with GRASP, even though the mean values obtained by SA outperformed those achieved by GRASP. These results lead us to select SA as local search algorithm to hybridize heuristics based on collective intelligence in order to improve the benchmarks of the problem of our interest.

References

1. Chun-Lung Chen,"An Iterated Local Search for Unrelated Parallel Machines Problem with Unequal Ready Times", Proceedings of the IEEE International Conference on Automation and Logistics Qingdao, China September 2008.
2. E. Ferretti and S. Esquivel, "Knowledge Insertion: An Efficient Approach to Simple Genetic Algorithms for Unrestricted for Parallel Equal Machines Scheduling". GECCO'05, 1587-1588, 2005, Washington DC, USA.
3. E. Ferretti and S. Esquivel, "An Efficient Approach of Simple and Multirecombinated Genetic Algorithms for Parallel Machine Scheduling", IEEE Congress on Evolutionary Computation, 1340-1347, September 2005, 2, Scotland, UK, IEEE Center.
4. E. Ferretti and S. Esquivel, "A Comparison of Simple and Multirecombinated Evolutionary Algorithms with and without Problem specific Knowledge Insertion, for Parrallel Machines Scheduling", International Transaction on Computer Science and Engineering, 2005, volume 3, number 1, 207-221.
5. E.G. Talbi, "Metaheuristics from design to implementation", by John Wiley & Sons, Canada, 2009.
6. Georgios C. Anagnostopoulos and Ghaith Rabadi,"A Simulated Annealing Algorithm for the Unrelated Parallel Machine Scheduling Problem", World Automation Congress Eight International Symposium on Manufacturing with Applications, Orlando, Florida, USA, June 9-13, 2002.
7. J. Beasley, OR-Library."http://people.brunel.ac.uk/mastjjb/info.html".
8. Joaquín Derrac, Salvador García, Daniel Molina, Francisco Herrera,"A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", Swarm and Evolutionary Computation, 2011.
9. Kai Li and Ba-Yi Cheng,"Variable neighborhood search for uniform parallel machine makespan scheduling problem with release dates", 2010 International Symposium on Computational Intelligence and Design.
10. M. Friedman,"The use of ranks to avoid the assumption of normality implicit in the analysis of variance", journal of American Statistical Association 3 (1937) 674-701
11. M. Friedman,"A comparison of alternative test of significance for the problem of the m rankings", Annals of Mathematical Statistics 11 (1940) 86-92.
12. Marc Sevaux, Kenneth Sörensen,"VNS/TS for a parallel machine scheduling problem", MEC-VNS: 18th Mini Euro Conference on VNS, 2005.
13. M. Pinedo,"Scheduling: Theory, Algorithms and System", Prentice Hall, 1995.
14. N. Metropolis and A. Rosenbluth and M. Rosenbluth and A. Teller, and E. Teller. "Equation of state calculations by fast computing machines". Journal of Chemical Physics, 21:10871092, 1953.
15. Panneerselvam Sivasankaran, Thambu Sornakumar, Ramasamy Panneerselvam, "Design and Comparison of Simulated Annealing Algorithm and GRASP to Minimize Makespan in Single Machine Scheduling with Unrelated Parallel Machines", Intelligent Information Management, 2010, 2, 406-416, doi:10.4236/iim.2010.27050 Published Online July 2010 (http://www.SciRP.org/journal/iim).
16. T. Bartz-Beielstein, "Experimental Research in Evolutionary Computation", The New Experimentalism, Springer, 2006.
17. T. Morton and D. Pentico,"Heuristic Scheduling Systems",John Wiley and Sons, New York, 1993.